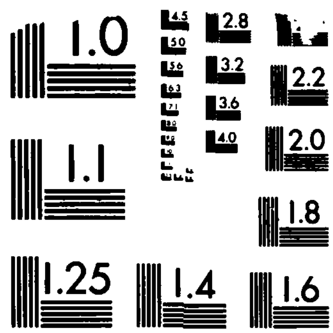
AD-A137 995 A TARGET PRIORITIZATION AID USING ARTIFICIAL INTELLIGENCE(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING J M DAVIS 06 DEC 83 1/1

UNCLASSIFIED AFIT/GCS/EE/83D-007 F/G 12/2 NL

END
FILMED
4-84
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.8
3.2
3.6
4.0

AIR FORCE INSTITUTE OF TECHNOLOGY

AIR UNIVERSITY
UNITED STATES AIR FORCE

A TARGET PRIORITIZATION AID
USING ARTIFICIAL INTELLIGENCE

THESIS

AFIT/GCS/EE/83D-007 Jonathan M. Davis

SCHOOL OF ENGINEERING

WRIGHT-PATTERSON AIR FORCE BASE, OHIO

AFIT/GCS/EE/83D-007

A TARGET PRIORITIZATION AID USING ARTIFICIAL INTELLIGENCE

THESIS

AFIT/GCS/EE/83D-007 Jonathan M. Davis
Captain USAF

Approved for public release; distribution unlimited.

AFIT/GCS/EE/83D-007

A TARGET PRIORITIZATION AID
USING ARTIFICIAL INTELLIGENCE

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by
Jonathan M. Davis, B.S.
Captain USAF
Graduate Computer Systems

December 1983

Acknowledgments

I thank my thesis advisor, Captain Robert Milne, for his careful consideration of my work and his helpful guidance throughout the this thesis effort. His suggestions on direction of research and his patience with my questions made my work more fruitful and enjoyable. I also thank my reader, Dr. J. Gary Reid, for his careful reading and probing critiques of my thesis drafts. Finally, I thank Sharon, Benjamin and Leanne, for their love and support throughout this thesis effort.

Contents

List of Figures

Abstract

In this thesis, an existing support aid for tactical offensive counterair targeting (the Target Prioritization Aid or TPA) is converted from an Operations Research structure to an Artificial Intelligence structure. The TPA was too limited in both its construct and its ability to support the tactical targeter. Requirements were generated that stressed ease of understanding, simplicity, and more human-like decision making. A general design was developed that addressed most of the requirements while keeping the basic functions of the TPA. Only one area, the inference engine, was designed in detailed and implemented. The other areas were represented by a skeleton system that provided an operating structure for testing purposes. This testing indicated that the rule-based production system developed in this thesis effort showed the capability of effectively replacing the TPA in all functional areas. In addition, its simplicity and growth potential make it easy to understand and improve.

A TARGET PRIORITIZATION AID USING ARTIFICIAL INTELLIGENCE

I. Introduction

Targeting aircraft against identified enemy targets with the intent of reducing their sortie generation capability has always been an integral part of Air Force tactical planning. The targeter, an Air Force officer, must know the capability and availability of the weapon systems at his command for offensive counter-air (OCA) strikes. Knowledge of the enemy's aircraft, base design, defensive posture, damage state, and numerical strength is also essential. But, the most important yet least tangible area of targeting is predicting the effectiveness of attacks on individual sites, determining the interaction between multiple targets, and building a cohesive optimal plan of attack involving numerous "friendly" aircraft and enemy targets.

The average tactical targeter has had formal training in all three of the above areas but little actual experience in the latter. In spite of this, many of the rules he uses to build his plan are not necessarily those learned in that training. In many cases, his environment is either unique or rapidly evolving. He is more likely to use heuristics passed on to him by a predecessor or derived from his own knowledge. This knowledge is a blend of targeting data,

experience, and his inate problem solving methodology. Since the last two are symbolic in nature, traditional data automation techniques encounter problems when attempting to represent a targeter's knowledge. (Callero, 1981)

In the last few years, data automation has been increasingly incorporated in the targeting process. Rome Air Development Center (RADC) is currently sponsoring a project with the intent of supporting the targeter. The Target Prioritization Aid (TPA) is being developed to explore the feasibility of assisting the tactical targeter in his decision making process. The purpose of the TPA is to recommend an optimal targeting plan to the user. (Tibbits, 1983) Targeting knowledge is represented numerically and the decision making process does not accurately model human targeters. This thesis project is aimed at improving upon that system through the use of a decision making technique that better models the human targeter and his knowledge.

Background

The TPA system is written in a non-standard version of APL, which is a high-level language with traditional data representation and control structures. It is a prototype written primarily for researching the potential of data automation in Air Force tactical targeting. As such, it only considers a small part of the entire targeting process and uses a representative subset of enemy bases, aircraft,

and base components. The heart of TPA is a series of linear functions based on Operations Research (OR) theory that calculates the benefit of an attack in terms of reductions in the enemy sortie generation and the cost of that attack in terms of number of aircraft required. A ratio of the benefit to cost is used to order the enemy targets into prioritized list. (Figgins, 1983) An extensive user interface exists to edit input data and report results, but the degree of freedom to change the decision making parameter is very limited. More on the TPA is presented in Chapter Two.

OR techniques have traditionally been applied to solving complex structured problems. The basic method, linear programming, mathematically models the problem by drawing relationships between selected input factors and a desired result. Instead of mapping the process a human problem solver would go through, the emphasis is on producing the same results as the system being modeled. This method guarantees an optimal solution within the constraints of the model, but an exhaustive search is necessary. Although the model is usually less complex than the problem, it is still not easily comprehended by an uninitiated user. An example application concerns solving an optimization problem commonly called the "travelling salesman" problem. The optimal solution is one which results in the shortest path the salesman must travel in order to visit all cities on his itinerary one and only one time. Chapter Two holds more on linear programming.

AI techniques have been used to solve unstructured problems in the same way that experts do. Some example applications are a production system that configures VAX computers (McDermott, 1982) and a diagnostic tool for determining the type of bacteria present in an infection (Davis, 1982). A rule-based production system models the way an expert makes decisions. It too uses heuristics to guide its decision process. This means decisions flows that are understandable to the average user and results that are comparable to human experts. In addition, the basic structure lends itself to easy modification and helpful feedback. A final strength of a production system is its capability to handle symbolic representation. (Winston, 1979) More on production systems is covered in Chapter Two.

Problem

The TPA has served its purpose as a research tool for investigating automated support for the expert targeter. However, its scope is limited due to deficiencies in the areas of ease of understanding, growth potential, and decision making. Either a new system or extensive modifications to this system must be generated to explore the vast potential of fully automated targeting.

The TPA is a complex and inflexible system due to the language (APL) and the decision making process (linear functions) used. To a non-technical user, the traditional form and flow of a language such as APL are both cryptic and

in-human. Since the version used is not standard, it would be unfamiliar to a outside analyst. Linear functions suffer from the same disadvantages due to their mathematical nature and iterative flow. This complexity and inflexibility makes the cost of additions or modifications prohibitive, because the analyst would need extensive development time.

The TPA is strictly a support tool that cannot be expected to do expert targeting in its present form or in future versions employing the same technology (OR). It provides no capacity for introducing alternate strategies beyond allowing weighting parameters on certain data elements to be changed. There is little intermediate control in the system: the initial parameters are set, the process started, and then no chance for changes until the whole process is done. Other than rationales for various weights, there is no help facility to answer questions such as how a decision was made or why an action was performed. The TPA does not model a targeter's decision making but instead draws a mathematical relation between the inputs and decisions that produces results similar to a targeter.

Scope

This thesis effort attempts to show that a production system based on AI theory can model a targeter while remaining easy to understand and modify. The solution domain was limited to AI on guidance from RADC. An expert system is designed that will provide all the capabilities of

the current system. The design also includes some basic strategy, an intermediate control point, and a primitive help facility. The implementation is confined to a skeleton system with only some of the basic decision making process fleshed out.

Approach

The first step was to analyze the TPA, the type of decision making (OR) it employed, and the type of decision making (AI) suggested as a replacement. The TPA was looked at for its purpose, achievements, and limitations. AI and OR were studied to find their applicability to the tactical targeting environment, as well as their advantages and disadvantages.

The second step was to determine the requirements. This was done by answering three questions. What was required to competently replace the TPA? What were the users needs that could be met within the current context? What was required to make the TPA an expert system?

Next, the overall design was generated. All functions performed by the TPA were covered. This included the user interface, the make up of the data bases, the decision making subsystem (or inference engine), and the strategy function.

In the fourth step, the design was expanded to include a detailed description of the inference engine. The control structure was determined and the system flow characterized.

Each sub process was functionally described.

At this point, a fully functioning skeleton system was implemented. The system was built modularly so that the full implementation could be easily done later. The sub processes in the inference engine were coded to provide a virtual copy of the existing system, demonstrate human-like decision making, and increase clarity and flexibility. Only enough coding was done to give a flavor of strategy and a help facility.

Finally, the performance of the new system was evaluated in three areas. How well was the system constructed in terms of speed, flexibility, and ease of use? How did it compare to the old TPA in terms of accuracy, complexity, and easiness to change? What potential did the help and strategy functions have?

Support

Two computer systems were used in support of this thesis project. All coding for the implementation of the skeleton system was done on the VAX 11/780 computer system at AFIT. The operating system was UNIX Berkeley Standard Version 4. An S-100 based personal computer system running the CP/M operating system was used for all text generation.

Two major software tools were also used in support of the project. The new system was coded in the production language OPS5 (Forgy, 1981). This is an interpreted language specifically developed for creating expert systems.

It was already available on the VAX. The software tool used for word processing was Word Star.

Organization

Each chapter of this paper represents a major phase of the research effort. The order of presentation coincides with the order in which each phase was accomplished. To become acquainted with the TPA, the OR field, and the AI field, an analysis in the form of a literature review was conducted. Chapter Two relates the results of this analysis and presents the requirements that were generated. Chapter Three consists of the general design for the entire system. The four functional areas described are user interface, data base, decision making, and strategy. One of these areas, decision making, is further developed for full implementation. Chapter Four details this additional design effort. The implementation of a skeleton system with fully functioning decision making is described in Chapter Five. The results are presented and analyzed in Chapter Six. The last chapter summarizes the effort and offers some recommendations.

II. Analysis and Requirements

This chapter relates the findings of the literature review in the analysis stage of the thesis effort. Some background information, example applications, and relative merits are presented for Operations Research (OR) and Artificial Intelligence (AI). The Target Prioritization Aid (TPA)is described and its limitations discussed. The results are then drawn upon to establish requirements for the proposed system.

Target Prioritization Aid

Rome Air Development Center (RADC) is currently sponsoring a project with the intent of supporting the targeter. The Target Prioritization Aid (TPA) is being developed to explore the feasibility of assisting the tactical targeter in his decision making process. It is a prototype written primarily for researching the potential of data automation in Air Force tactical targeting. The purpose of the TPA is to recommend an optimal targeting plan to the user. (Tibbits, 1983)

This plan is a prioritized list of component targets on enemy bases with matched friendly sortie requirements. The TPA only considers a small part of the entire targeting process and uses a representative subset of enemy bases, aircraft, and base components to generate the plan. It

makes decisions based on some linear functions developed through OR techniques with weights derived by polling experts. These are used to calculate a benefit value in terms of reductions to the enemy's potential sortie generation that is gained from attacking a specified component. A cost in terms of numbers of friendly sorties required is similarly obtained. The prioritization is done by ranking the components by their benefit to cost ratios. (Figgins, 1983) The TPA system is written in a non-standard version of APL, which is a high-level language with traditional data representation and control structures. Targeting knowledge is represented numerically.

Extensive user interface exists to edit input data and report results. The TPA does provide an optimal plan within the constraints of the simplified problem. Rather than being the approved solution, it is intended to be a guideline for the user. By providing two other plans, the TPA also bounds the problem. These plans are the result of holding either benefit or cost constant while maximizing the other. The TPA provides a beneficial side-effect by having the Air Order of Battle (AOB) and the Air Installation File (AIF) online. To the user, this is an improvement over hard copy. (Adelman, 1983)

A number of limitations are apparent in the TPA. The decision making process does not accurately model human targeters. Because it is based on OR techniques, it is difficult to understand. Even maintenance programmers can

encounter difficulties with the cryptic APL programming language (a non-standard version too). This makes any modifications or additions to any part of the system quite a task. The decision making function is doubly difficult to change due to the fragile nature of the integrity of the formulas. Because the problem had to be simplified and the decision parameters are not easily modified reflect the user's method, the value of this aid is limited. It can only be used as a very inexact questionable measuring stick to guide the user. The user can feel no confidence in the ability of the system to generate plans comparable to his own. (Adelman, 1983)

Operations Research

OR techniques have been applied in the modeling of human decision making and finding optimal solutions to well defined problems. The basic method, linear programming, mathematically models the problem by drawing relationships between selected input factors and a desired result. These relationships are in the form of linear equations that are solved in an iterative manner.

One application concerns an optimization problem commonly called the "travelling salesman" problem. (Thierauf, 1970) In this problem, a salesman must visit several cities in his region. There is no priority between any of the cities, so the salesman may visit them in any order. However, he may travel to each city just once. The

optimal solution is one which results in the shortest total distance traveled. For small numbers of cities, an exhaustive search or a linear programming technique can determine the optimal solution. Larger numbers require excessive computer time or the use of heuristics.

There are good points to using OR techniques in a decision making system. An OR based system produces the same results as the system being modeled, and can be guaranteed to find the optimal solution if it exists. The theory behind a system can be proven to do what it was intended to do. (Thierauf, 1970) This theory and the resultant formulas can be complex, but the development costs related to data processing are low. The model is usually less complex than the problem.

There are also disadvantages to employing OR techniques. The search for the optimal solution is exhaustive. For some problems, this means unexceptable run times or memory requirements. Both the theory and the resultant formulas are hard for the user to comprehend. (Milne, 1983) This lack of understanding can prevent the user from fully utilizing the system. The complex interrelations of the formulas hinders any attempts to modify or add to the system. Unlike humans, OR systems represent data and knowledge as numbers and equations. In addition, it does not map the process a human problem solver would go through.

Artificial Intelligence

AI techniques have been used to solve problems that are unstructured and normally solved by experts. The intent is to solve the problem in the same manner as the expert. This is done through the use of rule-based production systems. The rules are simple IF-THEN constructs that perform the specified actions in the THEN portion when all the conditions in the IF portion are true. The flow within the system is not linear as in traditional programs. The expert knowledge in the system is contained in the rules instead of in data. (Winston, 1979)

One application is a production system, R1, that configures VAX computers (McDermott, 1982). This production system, developed by Digital Equipment Corporation, is an operational system that has been extensively tested and verified before being put to use in place of a human expert. Given an order from a customer, it checks for missing parts, configures a workable VAX system, and determines any extra parts that may be needed to fully realize that system. Another expert system, MYCIN, acts as a diagnostic tool for doctors who are not specialists (Davis, 1982). It is an interactive system that determines the type of bacteria present in an infection. Apart from the system's expertise, the most notable achievement is its extensive help facility that explains both how and why conclusions were drawn.

Many advantages lie in using a production system. It models the way an expert makes decisions by using heuristics

to guide its decision process. Because of this, it is very human-like in its flow towards a decision and easy for a user to understand. Th user is therefore able to maximize the usefulness of the system. This coupled with the simplicity of the constructs means that modifications and additions are not difficult. The search process is not exhaustive, so the solution of large complex problems is possible. The results are comparable to human experts. Because the data is represented symbolically, the user can better relate to the process. (McDermott, 1982) A final strength of a production system is its capability to handle symbolic representation. (Winston, 1979)

Some disadvantages exist with production systems. The optimal solution cannot be guaranteed. However, there is a good chance of finding an optimal solution, and in many cases an optimal solution is not required. The system cannot be proven to do exactly what it is intended to do. In applications that are not well defined, this is not possible for any technique. The most serious drawback to production systems is their huge development costs. The low end of the spectrum for past efforts has been on the order of five man-years. (Davis, 1982)

Requirements

The resulting requirements fall in three categories. The first contains those basic requirements necessary to produce a virtual copy of the existing system through use of

any technique. The second set of requirements satisfies any users needs that cannot be met through the current technique. The last set necessitates using an AI technique.

The new system should perform the same functions with results comparable to or better than the TPA. It must determine the airbases and components pertinent to the given mission. The benefit accrued in attacking individual components at specified airbases must be calculated. The cost incurred by these attacks must also be calculated. A prioritized list of targets based on the cost vs. benefit trade off must be generated. This list must be presented as a whole or in fragments such as by day, by airbase, or by component. The two optimized plans for attack must be generated and displayed. The ordering of targets should be the same if given identical data and input parameters.

The new system should improve on the decision making while increasing clarity and flexibility. The logic used must be simple in nature but able to solve complex problems. Expert data should be the basis for all decision making. The decisions must be intuitive and traceable. Modifications to the decision making parameters of the system must be possible for the expert user without necessitating analyst intervention.

Three areas of enhancements are infusing the system with some strategy, allowing for growth, and providing a better help function. The new system must allow the consideration of such things as the affect of interaction

between components on sortie generation or interaction between airbases on self-defense from attack. Additional data items or types as well as new rules must be easily incorporated. The user should be able to query the system on such things as the logic path for a specified decision or the reasons for a given action.

III. Overall Design

This chapter covers the overall design of the system at a high level. Four major functions are discussed. The user interface function handles all portions of the system that involve communication between the user and the system. All input data, results, and control information are dealt with in the data base function. The inference engine is the heart of the decision making process. The last function, strategy, covers the setting of decision parameters.

User Interface

The user interface function consists of all rules and procedures that have any interaction with the user's terminal. This function can be further divided into four sub-functions. These sub-functions are: (1) allowing the user the ability to control various aspects of the system such as the strategy; (2) providing a facility to enter new information or additional data required for current calculations or decisions; (3) handling any questions or requests for help from the user; and (4) outputting the results of the session.

Four types of control accessible to the user during run time are incorporated. The user selects, from a standard set, the initial data bases to be used or may start with an empty set. The user may elect to constrain the decision

making process to a certain strategy at either the beginning or a predetermined breakpoint. At these same points, the user may indicate a desire to provide additional data. A final form of user control is to mark data objects for the purpose of either hiding those objects or insuring preferential treatment.

The user may input five types of additions to the current data base. New or updated enemy aircraft types may be added to the Air Installation File (AIF). New or updated enemy bases may be added to the Air Order of Battle (AOB). New or updated base components may also be added to the bases in the AOB. In response to system inquiries, the user must enter the weather conditions for the current day and the predicted weather for the rest of the attack period. All enemy aircraft considered threats to the combat zone and candidates for suppression must be entered.

The user is supported by three types of help. A time trace of all rules contributing to a decision is saved for both the user's understanding and for the analyst's debugging process when performing modifications. The system relates all conditions met in producing an action if the user asks why that action was performed. If the question is how a decision was arrived at, the system generates a partial trace involving only the rules that were immediate related predecessors to the specified decision.

The same type of result is provided in the first four output formats. The result is a list of targets ranked by

their benefit to cost ratios. The first format lists all the targets for an entire day. The other formats simply limit the extent of the list to by-base, by-component, and by aircraft. The last format is to output the pertinent values for individual targets.

Data Base

The data base consists of four general areas of data which contain both symbolically and numerically represented data. The two that are primarily the unchanging data set are the enemy data area and the user data area. The other two areas, control information and results, contain data elements that are modified during the course of the session.

The first general type in the enemy data area is the base structure which consists of such items as base name, BE number, and coordinates. The next structure, component, contains such information as damage states and capacity. The type and number of aircraft stationed at particular bases is the type of information in the aircraft structure.

The information that the user enters to set the general parameters of the session is found in the user data. This includes structures that hold data elements corresponding to the weather for a particular day or the type of aircraft the user wants to suppress.

The control information is used to monitor the state of the system and control both the flow of work and the decisions allowed. Information about the status of various

calculations is used to determine the progress of the session. Flags and status indicators are used to impose structure on the flow within the system. This control can serve to restrict activity to one group of rules or to select and switch from one to another. A strategy can be overlayed through the use of parameters that serve to advance certain rules while withdrawing others. Collections of specific settings of parameters developed by experts are contained in a expert data base.

The results area of the data base consists of three types of data which are represented either numerically or symbolically. Results of calculations, such as the benefit value derived for a target, are numerical in nature. The flags that indicate an object's eligibility for calculation or decision are represented symbolically. An example would be a flag set by the user to consider a certain aircraft. The trace output, that shows the decision making chain, is stored in ASCII format in a system file.

Inference Engine

The decision making portion of the system, also called an inference engine (Davis, 1980), is made of basic building blocks called rules and a methodology for determining which rule to do. Additional structure can be attained by employing a hierarchial grouping of rules. These groups are also be called contexts (McDermott, 1982).

Control is achieved by first imposing a hierarchial

structure to the rules. Rules are classified as being system level, context level, or member level. System level rules take precedence over all and either interact with the user or perform system wide functions such as handling trace data. Context level rules perform all context control functions such as marking a context active or terminating a context. The last level consists of individual rules with the least precedence which make up the contexts. Within a class, the user defined parameters are the next differentiation. At the bottom of the control structure is the standard mechanism of the selected implementation tool. This mechanism usually gives precedence to rules that have more conditionals than others, so the designer may engineer the rules to impose a order.

The inference engine has six major contexts that perform the basic functions for the user and for system maintenance. The first context is really a super context that has as its members the context level rules that manage the flow between contexts. Next is the group of rules that input all the enemy and user data. The strategy context contains the processes that accept and set the parameter values. All the rules that are used to calculate the benefit of attacking a target reside in the benefit context. A parallel context determines the cost. The last group accepts the user's choice on output type and presents the data.

Strategy

Strategy is imposed on the system through the setting if three types of decision parameters. These parameters included emphasizing the proximity of enemy bases to each other, stressing the importance of specific components, and highlighting certain facts for determining defense suppression costs. A close distance between enemy bases may enable more efficient defense suppression by friendly forces or provide the bases with convenient backup for vital resources. Certain components may be more critical to a base's effectiveness than others under certain conditions. An example is the control tower in bad weather. Defense suppression costs can be affected by such things as the clustering of enemy bases, the type of secondary aircraft, and the distance to those bases.

The parameters may be initialized from a selected expert data base or entered in a interactive mode by the user. This may be done at the beginning of the session or at any breakpoint. These breakpoints occur between the running of contexts and when the main benefit/cost calculation cycle is finished. This main cycle would have to be restarted or partially reversed depending on the extent of the new parameters.

IV. Inference Engine

This chapter details the low level design of only one portion of the system -- the inference engine. The inference engine is characterized by homogeneous groups of rules operating under a structure that controls flow between and within these groups. This control structure imposes a hierarchy on the rules with higher rules taking precedence over lower ones in case of conflict. In addition, a basis for choosing within a group is provided. The groups of rules are called contexts and include such processes as calculating the cost of attacking a target.

Control

The rules in the inference engine are arranged in the hierarchial order of system level, context level, and member level for control purposes. System level rules take precedence over all rules and either interact with the user or perform system wide functions. Context level rules perform all context control functions. Member level rules, the lowest in precedence, make up the contexts. When conflicts occurs between rules of the same level, additional mechanisms such as user parameters or rule size are used.

The system level, the top level of rules in the inference engine, includes rules that perform three functions. The first set handles all dialogue between the

user and the system. A second set does such system utilities as initialization, setting context status, opening files, and closing files. The last group allows and enforces user control over the direction of the session.

The middle level of rules, the context level, is made up of rules that control the actions of contexts. These rules determine the order in which the contexts will run and control flow both between and within contexts. When a context is opened, all other contexts are locked out by being closed. When that context terminates, the next eligible context is opened. Two or more related contexts may be open at the same time. A third set monitors the status of all contexts.

The last level of rules is called the member level because these rules simply are members of the various contexts. They perform low level calculations, actions, and decisions. Examples of these three functions are: (1) summing all the costs for a single enemy base; (2) marking a base for consideration because it is close enough to the combat zone; and (3) determining which of all the calculated benefit values is the highest.

To differentiate within a class, the control mechanism makes use of user defined parameters, size or complexity of rules, and the standard mechanism of the selected implementation tool. An example of a user defined parameter is one which precludes using any rules which solely consider distance of the candidate base from the combat zone. Only

those rules which use both distance from combat zone as well as distance from other enemy bases would be eligible for selection. First consideration is also given to those rules which appear more complex based on the number of conditionals that must be true. The final conflict resolution depends on the implementation tool. These usually give precedence to those rules whose conditionals are satisfied by the most recently changed data.

Contexts

Five primary contexts make up the basic inference engine. These five contexts input data, receive and set strategy parameters, determine the benefit of attacking a target, calculate the cost of attacking a target, and present output in the user specified format.

Three functions are done by the rules in the input context. The first function is to initialize the blank data structures with appropriate header information, read in all the initial data from the selected AOB and AIF data files, and fill the structures with that information. The second function is to receive and place new information from the user during the session. This new data may either modify an existing object or establish an entirely new object. The third function is to determine what additional data is required, have the appropriate system level rules get that data from the user, and place that data into the data base.

The strategy context is responsible for establishing

overriding controls by the user that influence the decision making. One set of rules within this context determines, through the appropriate system level rules, whether the user wishes to define his own parameters or select an expert data file. Another set inputs data from one of these data files or the user. The setting of these parameters is accomplished by the last set of rules.

Five steps must be performed in order to calculate the benefit of attacking a target (component). First, all bases with indicated aircraft are checked for distance from the combat zone (and possibly from other bases). The result of this check is used as an input into the second step -- finding the potential sortie generation rate for the component. Next, the cumulative damage effect is determined for the component. This is used, in the fourth step, to derive a residual which is an indication of how weakened the component is in terms of generating sorties. Through decrementing based on the residual, a figure is produced which relates to the number of enemy sorties that can be reduced by attacking the component.

In the current method of placing a cost on attacking a target, only two things are done. An initial fixed cost associated with enemy air defense suppression is assumed for each base that contains a target. This can be changed interactively by the user. The actual cost of attacking the individual target is determined through use of the component's capacity and an assessed sortie requirement

based on a prototype airbase.

Four groups of rules serve to generate the three types of output provided by the system. The first group finds out which type of output that the user desires. The second displays the benefits and costs for individual targets. This data is also broken down on the basis of base, component type, or aircraft type by the third group of rules. The fourth group presents the entire target list in prioritized order.

V. Implementation

This chapter begins with a rationale for the software tool that was picked (OPS5) and a description of that particular production language. The extent of implementation on all parts of the skeleton system except the inference engine is then covered. Next, the inference engine's implementation is detailed. The last section shows as an example the representative subset of rules which determines the distance factor.

OPS5

The selection of the AI software tool is based on both the requirements outlined in Chapter Two and what is either available on or compatible with the development computer. The requirements characterize a decision making system that is self-explanatory, human-like, and multi-faceted. Rule-based production systems fit these requirements. They are basically combinations of simple if-then statements that can yield complex results in a manner easily comprehended. OPS is currently available on the VAX and is a language for developing rule-based production systems. For a more detailed description than is offered below, consult the user's manual (Forgy, 1981).

The conditional statements that provide the decision making capacity in this language are called productions.

These productions are made up of two parts: the conditionals which are called the Left-Hand Side (LHS) and the actions which are called the Right_Hand Side (RHS). The LHS is a collection of patterns that must all be matched by at least one working memory representation. Only those elements in the representation corresponding to the elements in the pattern are considered. The RHS is a set of actions that are all performed when the LHS is true.

The working memory is primarily made up of descriptions of objects and relations among objects. These descriptions are called attribute-value representations and are groups of attribute-value pairs with an associated time tag. The values are scalar and may be either numeric or symbolic. The time tags are used to differentiate between the recency of elements as an aid to conflict resolution. Another type of representation is the vector representation which is used to hold sequences of symbols. The length of the vectors can change but may not exceed 127 values.

The conflict resolution strategy consists of five steps. First, all productions that have been performed (fired) are removed from the conflict set. This conflict set identifies all productions that have their LHS satisfied. Second, those productions whose first conditional is satisfied with the most recent working memory elements are selected as dominant. To choose from the productions which dominate, the remaining conditionals are compared for recency of their information. The most recent

dominates. The fourth step continues the tie breaking by determining the number of positive conditions that must be met in each LHS of the dominant set. The edge goes to the rule with the most. If all else fails, the fifth step arbitrarily choses one production to be the dominating rule.

Skeleton System

The system, as implemented in this effort, consists for the most part of only sketchily done portions. The user interface, data base, and strategy areas are not complete. These areas contain only enough rules and structures to give a flavor of a full operating system and to provide a working framework within which the basic decision making can occur. All basic functions of the inference engine are implemented. While some contexts are only sketchily done, most are relatively complete implementations. A more detailed discussion of the inference engine is contained in the next section.

The extent of user interface provided is some limited control, an overall trace, and two output types. The control consists of the ability to define a small number of parameters such as proximity of one base to another, to set such calculation values as initial defense suppression costs, and to select an output type. The trace is currently sent to the terminal for development purposes but could be diverted to a file. The output options are the single list of all target ordered by their benefit to cost ratios or the

display of the values for a specified target.

The data bases for the system are currently in an artificially pre-packaged format that facilitates their input. In the future, the real data would be put into these formats by a group of preprocessor rules. The enemy data is presented in three structures: bases, which contains very little initial data but storage elements for totals; aircraft-on-base, which has a representation for each type of aircraft for each base and contains distance and sortie-rate values; and components, which included such information as maximum damage effect and component capacity. All normal user input is placed into a structure called user that contains things like weather or into a structure called aircraft-selected which simply has an aircraft type. The only structure for control information is called control and contains all context status data.

By default, the strategy of the system resembles that of the old TPA system. No methods of employing expert strategy have been implemented beyond the user's input of a few parameters.

Inference Engine

The inference engine is sufficiently finished to provide comparable results to the old TPA. Control within the system is fully implemented. All contexts contain working rules. Some of the proposed enhancements are represented by only one or two rules which show the

possibilities. The basic benefit/cost derivation is fully supported. Only a small subset of the large amount of output generated by TPA is supported.

The entire hierarchy of rules including system, context, and member level is represented. The precedence of system over context level and context over member level is maintained. Control based on complexity of rule is also used. The conflict resolution employed through OPS5 is the MEA strategy which was described above.

Two contexts are only implemented as representative subsets of the old TPA's capability. The input context can be considered complete if the artificiality of the format of the data is ignored. This context would have to be enlarged to handle the raw data or an additional context must be created that would format the raw data. Only a small subset of the old TPA's output is presented by the output context. The formating is not pretty, since the main purpose of the output was for verification and debugging.

Three contexts not only fully represent the TPA capabilities but also contain enhancements. The benefit context currently provides the identical basis for determining benefit as is found in the old TPA. A parameter that places emphasis on the importance of a component is included. The cost context incorporates the ability to have the user change the initial fixed cost in addition to calculating cost in the same manner as the TPA. The last context, strategy, is in its entirety an improvement over

the TPA. It contains those rules which set the proximity and fixed cost parameters.

Distance Factor Example

This section presents the detailed explanation and description of two rules which calculate the distance factor. This factor is used to determine how many sorties can be maintained over a period of a day by the candidate base. The distance factor can assume one of three ranges of values: zero if the base is out of range for the specified aircraft; one if the base is close enough to the combat area that its frequency in sending out sorties of the specified aircraft is not reduced; and between zero and one if the frequency must be curtailed due to distance from the combat area (even with support).

The first production (Figure 1 and identified as "dist_rule_1" in the program listing in the Appendix) fires if a base is able to maintain a normal sortie generation rate. The LHS uses values that show the base's distance from the combat zone, the maximum distance at which the specified type of aircraft can still maintain the normal rate, and whether this determination has not been done for this aircraft/base combination. If the last is true and the distance from the combat zone is less than the maximum maintaining distance, the LHS is satisfied. The resulting actions change the distance factor value to one and set the flag to prevent any recalculations.

```
                    production DIST RULE 1

If there is
    in any CONTROL structure
        a data item CONTEXT NAME with value DISTANCE
        a data item CONTEXT STATUS with value OPEN
  and
    in any AIRCRAFT SELECTED structure
        a data item MODEL with value any NAME
  and
    in any AIRCRAFT structure
        a data item MODEL with value same NAME
        a data item MAINT SORTIE DIST with value X
  and
    in any AIRCRAFT ON BASE structure
        a data item MODEL with value same NAME
        a data item DIST FLAG with value "UNSTARTED"
        a data item DIST TO AREA with value less than X

Then
    in the selected AIRCRAFT ON BASE structure
        change data item DIST FLAG to value "STARTED"
        change data item DIST FACTOR to value 1
```

Figure 1. Distance Factor for Close Enemy Bases

production DIST RULE 2

```
If there is
    in any CONTROL structure
        a data item CONTEXT NAME with value DISTANCE
        a data item CONTEXT STATUS with value OPEN
 and
    in any AIRCRAFT SELECTED structure
        a data item MODEL with value any NAME
 and
    in any AIRCRAFT structure
        a data item MODEL with value same NAME
        a data item MAX SORTIE DIST with value X
        a data item MAINT SORTIE DIST with value Y
 and
    in any AIRCRAFT ON BASE structure
        a data item BASE with value any BASE NAME
        a data item MODEL with value same NAME
        a data item DIST FLAG with value "UNSTARTED"
        a data item DIST TO AREA with value greater than Y
            or less than X
 and
    in any COMPONENT structure
        a data item BASE with value same BASE NAME
        a data item IMPORTANCE with value YES

Then
    bind to A the computation (Z - X) / (Y - X)
and
    in the selected AIRCRAFT ON BASE structure
        change data item DIST FLAG to value "STARTED"
        change data item DIST FACTOR to value A
```

Figure 2. Distance Factor for Medium Range Bases

The second production (Figure 2 and identified as "dist_rule_2" in the program listing in the Appendix) fires if conditions exist that indicate that a base is not able to maintain a normal sortie generation rate but can maintain some minimal degraded sortie rate. The LHS uses values that show the base's distance from the combat zone, the maximum distance at which the specified type of aircraft can still maintain the normal rate, the maximum distance at which any rate can be generated, and whether this determination has not been done for this aircraft/base combination. The LHS is satisfied if the last is true and the distance from the combat zone is between the maximum maintaining distance and the maximum generation distance. The flag is set to stop any recalculations, and the distance factor is calculated. The value for the distance factor represents a sliding scale proportionate to where the base's distance lies in comparison to the maintaining distance and the generation distance.

The fact that a base is not near enough is determined by neither of the two previous productions firing. If no productions modify the element indicating that base's distance factor, then its value is zero since the default value is zero. A distance factor of zero makes all components on that base ineligible for targeting.

VI. Results

The results obtained from testing the new system are reported in this chapter. The system's performance in a stand-alone mode is analyzed first. Speed, flexibility, and ease of use are considered. Next, the new system is compared with the TPA for accuracy, complexity, and ease of change. The final section measures the effectiveness of the attempted enhancements. These added features are intermediate control points, some basic strategy, and a primitive help system.

Stand-Alone Performance

At this time, a large data base has not been used, so the amount of time needed to handle a sizable input is not known. However, results with a smaller data base have been quite good. The system processes approximately one tenth of the TPA's normal input in about a minute under ideal conditions. Because of its modularity and the relative independence of its rules, the new system is very flexible. Extensive restructing was accomplished during this project with minimal complications. Like the TPA, the new system is interactive and provides the user with simple menus and prompts. The user automatically receives information on which step is being performed and the status of the system. This all makes it easy to use.

TPA Comparison

Because the full data base has not been used, no comparision to the TPA for accuracy on an entire plan has been made. However, results from the limited input are consistent with the TPA. There is no comparision between the convoluted cryptic APL code in the TPA and the clear logical code of OPS-5. Peers with no experience in the latter find understanding the rules and the structure a simple task. They refused more than a quick glance at the former. As mentioned above, many changes were introduced to the system during the project. None of these changes entailed much time or effort with regard to programming. No attempts to modify the TPA code were made or considered.

Enhancements

The intermediate control points provided better support for the user by enabling more direct control over the process. Because of this, more attention could be focused on areas of concern. The strategy feature showed that analyst defined strategy could be easily used by the targeter, but that allowing the user to develop additional strategy interactively was beyond the scope of this effort. The problem is not difficult to solve within the AI framework however. Finally, the help facility did not provide satisfactory user support. It mainly functioned as an analyst's tool for debugging and testing purposes.

VII. Conclusion and Recommendations

This chapter presents a summary of the overall thesis project, the general results, and conclusions drawn from the course of the project. The results fall into the areas of the relative utility of the new system and the merits of AI as used in this system. Based on the results of the research and the experience garnered from this attempt, a few recommendations will be tendered.

Conclusion

The first step in this project was to formulate the problem. Basically, the TPA was too limited in both its construct and its ability to support the tactical targeter. Next a literature search was undertaken to gain some background into the TPA, linear programming, and expert production systems. This highlighted the limitations of both the TPA and linear programming while showing the potential of production systems. Requirements were generated that stressed ease of understanding, simplicity, and more human-like decision making. A general design was developed that addressed most of the requirements while keeping the basic functions of the TPA. Only one area, the inference engine, was designed in detailed and implemented. The other areas were represented by a skeleton system that provided an operating structure for testing purposes.

This testing indicated that the rule-based production system developed in this thesis effort showed the capability of effectively replacing the TPA in all functional areas. The amount of time this system took to make decisions was quite reasonable. In addition, its simplicity and flexibility made it easy to both understand and modify. In spite of this simplicity, relatively complex decisions were accurately made. The strategy and help features were demonstrated to be feasible but not difficult to implement. The actual programming was both simple to do and quickly done.

The problem addressed in this thesis effort was to show that a production system based on AI technology could model the way a targeter made decisions, while remaining easy to understand and modify. Intermediate control, strategy and help facilities were to be introduced. This thesis has shown that AI can be effectively used in the targeting environment, that the code is simple and flexible, and that changes are not difficult. Very basic examples of user control, strategy, and help features were implemented.

Recommendations

The development of support systems for the tactical targeter is an ongoing concern of RADC. Because this thesis project was sponsored by them to provide some insight into the possible future use of AI in the TPA area, some recommendations for further effort is in order. My specific

recommendations are that:

1) AI must be used as the basic technique for all decision making in the targeting environment.

2) Two man-years at a minimum must be planned for in order to implement a fully functioning useful system. This is due to difficulty in accumulating and refining the expert knowledge.

3) A structured form of the rule-based production system must be used. This move toward hybrid technique will help bridge the gap between the old method and the new.

4) The scope and purpose of the TPA be expanded to perform more decision making. This would be a step in the direction of fully automated targeting.

5) The TPA be moved to a LISP machine. As these machines become more hardened and less costly, their potential as a front line tool increases greatly.

Bibliography

Adelman, Leonard and Jackson Crowley. Target Prioritizaton Aid: Test Analysis Report. Contract #F30602-81-C-0263, Data Item: A006. New Hartford, New York: PAR Technology Corporation, August 1983.

Callero, M., D. Gorlin, F. Hayes-Roth, L. Jamison. Toward an Expert Aid for Tactical Air Targeting. Santa Monica, California: The Rand Corporation, January 1981.

Davis, Randall. "Meta-Rules: Reasoning about Control," Artificial Intelligence, 15 (3): 39-88 (December 1980).

Davis, Randall. "Expert Systems: Where Are We? And Where Do We Go From Here?," The AI Magazine, 3-22 (Spring 1982).

Figgins, Toini and Kerry Gates. Target Prioritization Aid: Functional Description. Contract #F30602-81-C-0263, Data Item A002. New Hartford, New York: PAR Technology Corporation, August 1983.

Forgy, Charles L. OPS5 User's Manual. Pittsburgh, Pennsylvania: Department of Computer Science, Carnegie-Mellon University, July 1981.

Gates, K. H., M. L. Donnell, L. Adelman, and J. O. Crowley. Test/Evaluation Plan for the Target Prioritization Aid. Contract #F30602-81-C-0263, Final Report. New Hartford, New York: PAR Technology Corporation, December 1982.

McDermott, John. "R1: A Rule-Based Configurer of Computer Systems," Artificial Intelligence, 19 (1): 39-88 (September 1982).

Milne, Robert. "Artificial Intelligence and Operations Research," presentation at the MORS Conference. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, October 1983.

Thierauf, Robert J. and Richard A. Grosse. Decision Making Through Operations Research. New York, New York: John Wiley & Sons, Inc., 1970.

Tibbits, Paul. Target Prioritization Aid System Specification Report. Contract #F30602-81-C-0263. New Hartford, New York: PAR Technology Corporation, April, 1983.

Winston, Patrick H. Artificial Intelligence. Reading, Massachusetts: Addison-Wesley Publishing Company, 1979.

APPENDIX

This appendix contains the program listing for the skeleton implementation. The following exchanges were made in the program to accomodate the printer: "[" for left brace; "]" for right brace; "*" for up-arrow; and "/" for absolute value.

```
;************ STRUCTURE AND ELEMENT DECLARATION ************

(vector-attribute
        data
        list_of_contexts)


(literalize input_output
        data)


(literalize list
        list_of_contexts)


(literalize control
                context_name
                context_status
                strategy
                stack_num)


(literalize system_variables
                run_status
                last_stack_num
                strategy)


(literalize airbase
                name
                BE_number
                capacity
                total_basic_sortie_rate)

(literalize aircraft_on_base
                base
                model
                dist_flag
                quantity
                dist_to_area
                dist_factor
                enemy_basic_sortie_rate)
```

```
(literalize component
                type
                base
                cum_eff_flag
                sortie_flag
                max_effect
                assessed_sortie_req
                proto_capacity
                importance
                capacity
                cumulative_effect
                component_residual
                sortie_req
                sortie_reduced
                ratio)


(literalize aircraft
                model
                sortie_gen_rate
                maint_sortie_dist
                quantity
                max_sortie_dist)


(literalize user
                init_fixed_cost)


(literalize aircraft_selected
                model)
```

```
;**************** SYSTEM LEVEL RULES **********************

        ;OPEN FILES AND SET RUN STATUS TO RUNNING

(p start_run
        (system_variables
                *run_status                 start)
-->
        (openfile
                tracefile                   /trace.tpa/ out)
        (openfile
                inputfile                   /data.tpa/ in)
        (openfile
                outputfile                  /out.tpa/ out)
        (default
                tracefile                   trace)
        (modify 1
                *run_status                 running))

          ;DISPLAY MAIN MENU AND RECEIVE OPTION

(p get_options
        (system_variables
                *run_status                 running
                *strategy                   <x>)
       -(control
                *context_status             open)
  [ <e> (list
                *list_of_contexts                   []) ]
-->
        (write
                (crlf) (crlf)
(crlf) (tabto 15) *****  FUNCTION  SELECTION  MENU  *****
                (crlf) (tabto 22) 1--Input Data from File
                (crlf) (tabto 22) 2--Input Data from User
                (crlf) (tabto 22) 3--Set Strategy
                (crlf) (tabto 22) 4--Calculate Benefit
                (crlf) (tabto 22) 5--Calculate Cost
                (crlf) (tabto 22) 6--Output results
                (crlf) (tabto 22) 7--Query database
                (crlf) (tabto 22) 8--Help
                (crlf) (tabto 22) 9--Done
  (crlf) (tabto 15) Please enter number of your choice-- )
        (bind
                <z>             (accept))
        (bind
                <z>             (compute <z> + 1))
        (make control
                *context_name               (substr <e> <z> <z>)
                *strategy                   <x>
                *context_status             candidate))
```

```
;***************** CONTEXT LEVEL RULES ********************

                    ;OPEN CANDIDATE CONTEXT

(p open_context
        (control
                *context_name                   <name>
                *context_status                 candidate)
       -(control
                *context_status                 open)
-->
        (modify 1
                *context_status                 open)
        (write
                (crlf) <name> context started))

                ;SPECIAL OPEN FOR INPUT CONTEXT

(p open_input_context
        (control
                *context_name                   input
                *context_status                 candidate)
       -(control
                *context_status                 open)
-->
        (modify 1
                *context_status                 open)
        (make input
                *data                           cont)
        (write
                (crlf) input context started))

        ;PUT CANDIDATE IN STACK IF ONE ALREADY OPEN

(p stack_context
        (control
                *context_name                   <name>
                *context_status                 candidate)
        (control
                *context_status                 open)
        (system_variables
                *last_stack_num                 <x>)
-->
        (bind <y>
                (compute                        <x> + 1))
        (modify 1
                *context_status                 stacked
                *stack_num                      <y>)
        (modify 3
                *last_stack_num                 <y>)
        (write  (crlf) <name> context stacked))
```

;START CONTEXT FROM STACK

```
;(p start_stack
        (control
                *context_name                    <name>
                *context_status                  stacked
                *stack_num                       [<y> > 0])
       -(control
                *context_name                    <> <name>
                *context_status                  open)
       -(control
                *context_name                    <> <name>
           *stack_num                    [<z> > <y> > 0])
-->
        (modify 1
                *context_status                  open
                *stack_num                       0)
        (write
                (crlf) <name> context started from stack))
```

;CLOSE CONTEXT THAT IS DONE

```
(p close_context
        (control
                *context_name                    <name>
                *context_status                  open)
-->
        (write
                (crlf) <name> context closed)
        (remove 1))
```

```
;******************** INPUT CONTEXT ********************

          ;READ ONE LINE OF DATA FROM INPUT FILE

(p read_data
        (control
                *context_name               input
                *strategy                   []
                *context_status             open)
       -(input
                *data                       eof)
        (input
                *data                       cont)
-->
        (remove 2)
        (make input
                (acceptline                 inputfile eof)))

        ;STORE BASE INFORMATION IN STRUCTURE AIRBASE

(p store_base_data
        (control
                *context_name               input
                *strategy                   []
                *context_status             open)
   [<e> (input
                *data                       base)]
-->
        (make airbase
                *name                       (substr <e> 3 3)
                *capacity                   0
                *total_basic_sortie_rate   0
                *BE_number                  (substr <e> 4 4))
        (modify 2
                *data                       cont))

      ;STORE INFORMATION IN STRUCTURE AIRCRAFT_ON_BASE

(p store_aob_data
        (control
                *context_name               input
                *strategy                   []
                *context_status             open)
   [<e> (input
                *data                       aob)]
-->
        (make aircraft_on_base
                *base                       (substr <e> 3 3)
                *model                      (substr <e> 4 4)
                *dist_flag                  unstarted
                *enemy_basic_sortie_rate 0
```

```
                *quantity               (substr <e> 5 5)
                *dist_to_area           (substr <e> 6 6))
        (modify 2
                *data                   cont))

         ;STORE INFORMATION IN STRUCTURE AIRCRAFT

(p store_aircraft_data
        (control
                *context_name           input
                *strategy               []
                *context_status         open)
   [<e> (input
                *data                   aircraft)]
-->
        (make aircraft
                *model                  (substr <e> 3 3)
                *sortie_gen_rate        (substr <e> 4 4)
                *maint_sortie_dist      (substr <e> 5 5)
                *max_sortie_dist        (substr <e> 6 6))
        (modify 2
                *data                   cont))

      ;STORE INFORMATION IN STRUCTURE AIRCRAFT_SELECTED

(p store_as_data
        (control
                *context_name           input
                *strategy               []
                *context_status         open)
   [<e> (input
                *data                   as)]
-->
        (make aircraft_selected
                *model                  (substr <e> 3 3))
        (modify 2
                *data                   cont))

        ;STORE INFORMATION IN STRUCTURE COMPONENT

(p store_comp_data
        (control
                *context_name           input
                *strategy               []
                *context_status         open)
   [<e> (input
                *data                   comp
                *4                      <name>)]
        (airbase
                *name                   <name>
                *capacity               <q>)
```

```
-->
        (make component
                *type                        (substr <e> 3 3)
                *base                        (substr <e> 4 4)
                *cum_eff_flag                unstarted
                *sortie_flag                 unstarted
                *assessed_sortie_req         (substr <e> 5 5)
                *proto_capacity              (substr <e> 6 6)
                *importance                  (substr <e> 7 7)
                *capacity                    (substr <e> 8 8)
                *max_effect                  (substr <e> 9 9)
                *component_residual          1
                *sortie_req                  0
                *sortie_reduced              0
                *ratio                       0
                *cumulative_effect           (substr <e> 10 10))
        (modify 2

                *data                        cont)
        (bind <a>
                (substr <e> 8 8))
        (bind <q>
                (compute                     <a> + <q>))
        (modify 3
                *capacity                    <q>))
```

```
;******************** USER CONTEXT *********************

                                  ;OBTAIN DATA FROM USER

(p input_from_user
        (control
                *context_name           user
                *strategy               []
                *context_status         open)
-->
        (write
                (crlf)  Not implemented))
```

```
;****************** STRATEGY CONTEXT **********************

          ;PRESENT AND ACCEPT OPTIONS FOR STRATEGY

(p ask_strategy
        (control
                *context_name                   strategy
                *context_status                 open)
-->
        (write
                (crlf)(crlf)
                (crlf) (tabto 20) STRATEGY OPTIONS
                (crlf) (tabto 17) 1--Consider importance
                (crlf) (tabto 17) 2--Set initial fixed cost
                (crlf) (tabto 17) 3--No strategy
        (crlf) (tabto 15) Please enter your option-- )
        (make input
                (accept)))

                      ;SET NO STRATEGY

(p set__nil
        (control
                *context_name                   strategy
                *context_status                 open)
        (system_variables
                *strategy                       [])
        (input
                *data                           3)
-->
        (modify 2
                *strategy                       none)
        (remove 3))

          ;SET CONSIDERATION OF IMPORTANT COMPONENT

(p set__importance
        (control
                *context_name                   strategy
                *context_status                 open)
        (system_variables
                *strategy                       [])
        (input
                *data                           1)
-->
        (modify 2
                *strategy                       importance)
        (remove 3))
```

;SET INITIAL FIXED COST FOR SUPRESSION

```
(p set_init_cost
        (control
                *context_name           strategy
                *context_status         open)
        (input
                *data                   2)
-->
        (write
                (crlf) Enter fixed cost-- )
        (make user
                *init_fixed_cost        (accept))
        (remove 2))
```

```
;**************** BENEFIT CONTEXT ********************

        ;SELECT CLOSE RANGE / COMPUTE DISTANCE FACTOR

(p dist_rule_1
        (control
                *context_name               benefit
                *strategy                   []
                *context_status             open)
        (aircraft_selected
                *model                      <name>)
        (aircraft_on_base
                *model                      <name>
                *dist_flag                  unstarted
                *dist_to_area               <x>)
        (aircraft
                *model                      <name>
                *maint_sortie_dist          [<y> >= <x>])
    -->
        (modify 3
                *dist_flag                  started
                *dist_factor                1))

        ;SELECT DEGRADED RANGE / COMPUTE DISTANCE FACTOR

(p dist_rule_2
        (control
                *context_name               benefit
                *strategy                   []
                *context_status             open)
        (aircraft_selected
                *model                      <name>)
        (aircraft
                *model                      <name>
                *max_sortie_dist            <x>
                *maint_sortie_dist          <y>)
        (aircraft_on_base
                *model                      <name>
                *dist_flag                  unstarted
                *dist_to_area               [<z> < <x>  > <y>])
    -->
        (bind <a>
         (compute               ((<z> - <x>) // (<y> - <x>)))
        (modify 4
                *dist_flag                  started
                *dist_factor                <a>))
```

```
          ;SELECT DEGRADED RANGE / COMPUTE DISTANCE FACTOR
          ;        USE IMPORTANCE STRATEGY

(p dist_rule_2s
          (control
                  *context_name             benefit
                  *strategy                 importance
                  *context_status           open)
          (aircraft_selected
                  *model                    <name>)
          (aircraft
                  *model                    <name>
                  *max_sortie_dist          <x>
                  *maint_sortie_dist        <y>)
          (aircraft_on_base
                  *base                     <base>
                  *model                    <name>
                  *dist_flag                unstarted
                  *dist_to_area             [<z> < <x>   > <y>])
          (component
                  *base                     <base>
                  *importance               yes)
      -->
          (bind <a>
           (compute                 ((<z> - <x>) // (<y> - <x>)))
          (modify 4
                  *dist_flag                started
                  *dist_factor              <a>))


          ;COMPUTE SORTIE-RATE FOR AIRCRAFT ON BASE

(p sortie_rate_1
          (control
                  *context_name             benefit
                  *strategy                 []
                  *context_status           open)
          (aircraft
                  *model                    <name>
                  *sortie_gen_rate          <x>)
          (aircraft_on_base
                  *model                    <name>
                  *quantity                 <y>
                  *dist_factor              [<z> > 0])
      -->
          (bind <a>
                  (compute                  <x> * <y> * <z>))
          (modify 3
                  *dist_factor              0
                  *enemy_basic_sortie_rate <a>))
```

```
        ;SUM SORTIE-RATE FOR BASE

(p sortie_rate_2
        (control
                *context_name                 benefit
                *strategy                     []
                *context_status               open)
        (aircraft_on_base
                *base                         <name>
                *enemy_basic_sortie_rate [<x> > 0])
        (airbase
                *name                         <name>
                *total_basic_sortie_rate <y>)
    -->
        (bind <y>
                (compute                      <x> + <y>))
        (modify 2
                *enemy_basic_sortie_rate  0)
        (modify 3
                *total_basic_sortie_rate <y>))

        ;COMPUTE COMPONENT RESIDUAL

(p residual_2
        (control
                *context_name                 benefit
                *strategy                     []
                *context_status               open)
        (component
                *cumulative_effect            [<x> > 0]
                *max_effect                   <y>)
    -->
        (bind <a>
                (compute                      1 - (<x> * <y>)))
        (modify 2
                *cumulative_effect            0
                *component_residual           <a>))

        ;COMPUTE SORTIES REDUCED BY COMPONENT

(p sorties_reduced_1
        (control
                *context_name                 benefit
                *strategy                     []
                *context_status               open)
        (airbase
                *capacity                     <c1>
                *total_basic_sortie_rate [<x> > 0])
        (component
                *capacity                     <c2>
                *component_residual           [<y> < 1])
    -->
```

```
(bind <a>
 (compute       (<x> * <c1> // <c2>) * (1 - <y>)))
(modify 3
        *component_residual     1
        *sortie_reduced         <a>))
```

```
;********************** COST CONTEXT *******************

        ;COMPUTE COST BY COMPONENT

(p cost_1
        (control
                *context_name           cost
                *strategy               []
                *context_status         open)
        (component
                *sortie_flag            unstarted
                *assessed_sortie_req    [<x> > 0]
                *proto_capacity         [<y> > 0]
                *capacity               [<z> > 0])
        (user
                *init_fixed_cost        [<k> > 0])
   -->
        (bind <a>
        (compute                (<x> // <y> * <z>) + <k>))
        (modify 2
                *sortie_flag            started
                *sortie_req             <a>))


        ;CALCULATE RATIO OF BENEFIT TO COST FOR COMPONENTS

(p calculate_ratio
        (control
                *context_name           cost
                *strategy               []
                *context_status         open)
        (component
                *sortie_reduced         [<x> > 0]
                *sortie_req             [<y> > 0]
                *ratio                  0)
   -->
        (bind <a>
                (compute                <x> // <y>))
        (modify 2
                *ratio                  <a>))
```

```
;GET VALUE FOR INITIAL FIXED COST IF NOT ALREADY

(p missing_init_value
        (control
                *context_name           cost
                *strategy               []
                *context_status         open)
        (user
                *init_fixed_cost        0)
    -->
        (write
        (crlf) please enter value for init fixed cost--)
        (modify 2
                *init_fixed_cost        (accept)))
```

```
;****************** OUTPUT CONTEXT **********************

    ;OUTPUT TO OUTPUT FILE THE RANKED LIST OF COMPONENTS

(p output_all_ranked
        (control
                *context_name           output
                *strategy               []
                *context_status         open)
        (component
                *type                   <type>
                *base                   <name>
                *sortie_reduced         <x>
                *sortie_req             <y>
                *ratio                  [<z> > 0])
       -(component
                *ratio                  [<a>  > <z>])
-->
        (write outputfile (crlf) <type> <name> reduced-- <x>
                        required-- <y>  ratio-- <z>)
        (modify 2
                *ratio                  0))


                ;MAKE COST A CANDIDATE FOR OPENING
                    ;TO RESET ERASED VALUES

(p restart_cost
        (control
                *context_name           output
                *strategy               []
                *context_status         open)
-->
        (make control
                *context_name           cost
                *context_status         candidate))
```

```
;******************** QUERY CONTEXT **********************

                    ;ASK AND RECEIVE OPTIONS

(p ask_query_type
        (control
                *context_name               query
                *strategy                   []
                *context_status             open)
-->
        (write
                (crlf)(crlf)
                (crlf) (tabto 20) QUERY OPTIONS
                (crlf) (tabto 17) 1--Display component info
                (crlf) (tabto 17) 2--Display aircraft info
                (crlf) (tabto 15) Please enter option-- )
        (make input
                (accept)))

                 ;GET COMPONENT AND BASE NAMES

(p query_comp_1
        (control
                *context_name               query
                *strategy                   []
                *context_status             open)
        (input
                *data                       1)
-->
        (write
   (crlf) Enter component name with base identifier-- )
        (modify 2
                *data                       (acceptline)))

                  ;PRINT COMPONENT INFORMATION

(p query_comp_2
        (control
                *context_name               query
                *strategy                   []
                *context_status             open)
        (input
                *data                       <type>
                *3                          <name>)
        (component
                *type                       <type>
                *base                       <name>
                *sortie_reduced             <tsr>
                *sortie_req                 <tsrq>
                *ratio                      <r>)
-->
```

```
        (write
                (crlf)   <type>        <name>
                (crlf) sorties reduced-- <tsr>
                (crlf) sorties req-- <tsrq>
                (crlf) benefit/cost-- <r>)
        (remove 2))

                        ;GET AIRCRAFT TYPE

(p query_aircraft_1
        (control
                *context_name           query
                *strategy               []
                *context_status         open)
        (input
                *data                   2)
-->
        (write
                (crlf) What aircraft do you want to see--)
        (modify 2
                *data                   (accept)))

                ;PRINT AIRCRAFT INFORMATION

(p query_aircraft_2
        (control
                *context_name           query
                *strategy               []
                *context_status         open)
        (input
                *data                   <model>)
        (aircraft
                *model                  <model>
                *sortie_gen_rate       <sgr>
                *maint_sortie_dist      <msd>
                *max_sortie_dist        <mxsd>)
-->
        (write
                (crlf)          <model>
                (crlf) sortie rate-- <sgr>
                (crlf) maint dist-- <msd>
                (crlf) max dist-- <mxsd>)
        (remove 2))
```

```
;******************* HELP CONTEXT *************************

                         ;DIRECT TO TRACE FILE

(p trace_info
        (control
                *context_name           help
                *strategy               []
                *context_status         open)
-->
        (write
                (crlf)  Trace is in §jdavis/ths/trace.tpa))
```

```
;****************** DONE CONTEXT ************************

                ;CLOSE FILES AND HALT RUN

(p session_done
        (control
                *context_name           done
                *strategy               []
                *context_status         open)
-->
        (write
                (crlf)(crlf)
                (crlf) (tabto 20) *****RUN FINISHED*****)
        (closefile
                tracefile
                inputfile
                outputfile)
        (halt))
```

```
;********************* INITIALIZATION ***********************

(strategy mea)
(watch 1)
(make system_variables
         *run_status              start
        *last_stack_num           0
        *strategy                 none)
(make user
        *init_fixed_cost          0)
(make list
         input user strategy benefit cost output query help done)
(run)
```

VITA

Jonathan Mark Davis was born on 19 July 1956 in Bismarck, North Dakota. Between the age of one and eleven, he lived for eight years in what is now Papua New Guinea. He graduated from high school in Mattoon, Illinois in 1974 and accepted an appointment to the United States Air Force Academy, Colorado Springs, Colorado. In May 1978, he received the degree of Bachelor of Science in Computer Science and was commisioned into the United States Air Force. He served at Headquarters SAC as a systems analyst and section chief on a large-scale Honeywell computer until entering the School of Engineering, Air Force Institute of Technology, in June 1982.

Permanent address: 714 Benedict
Chillicothe, Illinois 61523

unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
AFIT/GCS/EE/83D-7

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Air Force Institute of Technology

6b. OFFICE SYMBOL *(If applicable)*
AFIT/EN

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS *(City, State and ZIP Code)*
Wright-Patterson AFB, Ohio, 45433

7b. ADDRESS *(City, State and ZIP Code)*

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State and ZIP Code)*

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

11. TITLE *(Include Security Classification)*
TARGET PRIORITIZATION AID (U)

12. PERSONAL AUTHOR(S)
Davis, Jonathan M.

13a. TYPE OF REPORT
MS Thesis

13b. TIME COVERED
FROM Jun 82 TO Dec 83

14. DATE OF REPORT *(Yr., Mo., Day)*
1983 Dec 6

15. PAGE COUNT
75

16. SUPPLEMENTARY NOTATION

Approved for public release: IAW AFR 190-17.
LYNN E. WOLAVER 7 Feb 84
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Artificial Intelligence
Operations Research
Tactical Targeting

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

In this thesis, an existing support aid for tactical offensive counterair targeting (the Target Prioritization Aid or TPA) is converted from an Operations Research structure to an Artificial Intelligence structure. The TPA was too limited in both its construct and its ability to support the tactical targeter. Requirements were generated that stressed ease of understanding, simplicity, and more human-like decision making. A general design was developed that addressed most of the requirements while keeping the basic functions of the TPA. Only one area, the inference engine, was designed in detailed and implemented. The other areas were represented by a skeleton system that provided an operating structure for testing purposes. This testing indicated that the rule-based production system developed in this thesis effort showed the capability of effectively replacing the TPA in all functional areas.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐

21. ABSTRACT SECURITY CLASSIFICATION

22a. NAME OF RESPONSIBLE INDIVIDUAL
Robert W. Milne, Cpt, USA

22b. TELEPHONE NUMBER *(Include Area Code)*
513-255-5533

22c. OFFICE SYMBOL
ENG

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

unclassified

SECURITY CLASSIFICATION OF THIS PAGE